How to translate a book within an hour - Towards general purpose programmable human computers with CrowdLang

Minder, Patrick ; Bernstein, Abraham

Abstract: In this paper we present the programming language and framework CrowdLang for engineering complex computation systems incorporating large numbers of networked humans and machines agents. We evaluate CrowdLang by developing a text translation program incorporating human and machine agents. The evaluation shows that we are able to simply explore a large design space of possible problem solving programs with the simple variation of the used abstractions. Furthermore, an experiment, involving 1918 different human actors, shows that the developed mixed human-machine translation program significantly outperforms a pure machine translation in terms of adequacy and fluency whilst translating more than 30 pages per hour and that the program approximates the professional translated gold-standard to 75% using the automatic evaluation metric METEOR. Last but not least, our evaluation illustrates that our new human computation pattern staged-contest with pruning outperforms all other refinements in the translation task.

DOI: https://doi.org/10.1145/2380718.2380745

How to Translate a Book Within an Hour

Towards General Purpose Programmable Human Computers with CrowdLang

Patrick Minder
Dynamic and Distributed Information Systems Group
University of Zurich
minder@ifi.uzh.ch

Abraham Bernstein
Dynamic and Distributed Information Systems Group
University of Zurich
bernstein@ifi.uzh.ch

ABSTRACT

In this paper we present the programming language and framework *CrowdLang*[1] for engineering complex computation systems incorporating large numbers of networked humans and machines agents. We evaluate *CrowdLang* by developing a text translation program incorporating human and machine agents. The evaluation shows that we are able to simply explore a large design space of possible problem solving programs with the simple variation of the used abstractions. Furthermore, an experiment, involving 1918 different human actors, shows that the developed mixed human-machine translation program significantly outperforms a pure machine translation in terms of adequacy and fluency whilst translating more than 30 pages per hour and that the program approximates the professional translated gold-standard to 75% using the automatic evaluation metric METEOR. Last but not least, our evaluation illustrates that our new human computation pattern *staged-contest with pruning* outperforms all other refinements in the translation task.

Author Keywords

Human Computation, Crowdsourcing, CrowdLang

ACM Classification Keywords

H.1.2 User/Machine Systems : H.4.1 Workflow management

General Terms

Algorithms, Human Factors, Languages

INTRODUCTION

Much of the prosperity gained by the industrialization of the economy in the 18th century arose from the increased productivity by dividing work into smaller tasks performed by more specialized workers. Wikipedia, Google and other stunning success stories show that with the rapid growth of the World Wide Web, this concept of Division of Labour can also be applied on knowledge work [14, 13, 5]. These new modes of collaboration— whether they are called collective intelligence, human computation, crowdsourcing or social computing —are now able to routinely solve problems that would have been unthinkably difficult only a few years ago by interweaving the creativity and cognitive capabilities of networked humans with the efficiency and scalability of networked computers. The advent of crowdsourcing markets (e.g., Amazon's Mechanical Turk, Clickworker, or CrowdFlower) even fosters this development and Bernstein et al. suggest that, as the scale and scope of these human-computer networks increase, we can view them as constituting a kind of a "global brain" [5].

Even though there are hundreds of compelling human computation systems that harness the potential of this "global brain", our understanding of how to *"program"* these systems is still poor because human computers are different from traditional computers due to the huge motivational, error and cognitive diversity within and between humans [5]. As a consequence, today, human computation is mostly only used for massive parallel information processing for tasks such as image labeling or tagging. These tasks share in common that they are massively parallelizable, have a low interdependence between single assignments in the task decomposition, and use relatively little cognitive effort.

A plethora of tasks, however, cannot be captured in this paradigm. Consider, e.g., the joint editing of lengthy texts as accomplished on Wikipedia. Here, a large number of actors work on highly interdependent tasks that would be very difficult to cast into a bulk parallelization with low interdependence. Hence, to harness the full potential of human computation systems, we need new powerful programming metaphors that support the design and implementation of human computation systems, as well as general-purpose infrastructure to execute them. Specifically, we need a programming language that supports the whole range of possible dependencies between single tasks, allows for the seamless reuse of known human computation patterns incorporating both human and machine operators to exploit prior experience, and integrates multiple possible execution platforms such micro task markets as well as games-with-a-purpose platforms to leverage a large ecosystem of participants. Furthermore, to move from a culture of *"wizard of oz"*-techniques, in which applications are the result of extensive trial-and-error refinements, this programming language has to support the recombination [4] of interaction patterns to systematically explore the design space of possible solutions.

Recent research only partially addresses these challenges by providing programming frameworks and models [11, 10, 1]

[1]This research note is a short version of [15] in which we describe *CrowdLang* in more detail.

WebSci 2012, June 22–24, 2012, Evanston, Illinois, USA.

for massive parallel human computation, concepts for planning and controlling dependencies [3, 17], and theoretical deductive analysis of emergent collective intelligence [13].

In this article, we present the human computation programming language and framework *CrowdLang*. It supports cross-platform workforce integration, the management of latency caused by human computer, as well as incorporates abstractions for group decision, contest, and collaborative interaction patterns to exploit prior experience, as proposed by Malone et al. [13]. Furthermore, in contrast to several MapReduce inspired approaches [10, 1], *CrowdLang* supports the management of arbitrary dependencies among tasks and workers and not only synchronous parallelization. We illustrate *CrowdLang*'s feasibility and strength in interweaving human and machine actors by programming a collection of text translation programs. We show that these translation programs are capable to speedily translate non-trivial texts from German to English achieving a significantly better quality than pure machine translation approaches. In addition, given the simple recombination of patterns supported by *CrowdLang*, we were able to unearth a novel human computation pattern called *"Staged-Contest with Pruning"* that outperforms all other known patterns in the translation task.

As such, the contributions of the paper are as follows: *(1)* We present *CrowdLang* and the 'pattern recombinator' methodology – a translation of the process recombination approach [4] to human computation tasks. *(2)* We present a set of human computation programs that allow translating more than 30 pages per hour with a "good" quality when compared to professional translations. *(4)* Finally, we present the novel interaction pattern *"Staged-Contest with Pruning"* that outperforms all other patterns in the translation task.

BACKGROUND AND RELATED WORK

A number of programming frameworks and concepts that address the distinct challenges in engineering human computation systems were proposed recently. Little et al.'s [11] imperative programming framework *TurKit* incorporating iterative and parallel programing constructs in human computation. *Turkit* supports the idea of a *"crash-and-rerun"* programming model, which allows a programmer to repeatedly rerun algorithms without republishing costly previously completed human computation. Kittur et al.'s *CrowdForge* [10] is inspired by the MapReduce [9] programming model and describes human computation as a sequence of partitioning, mapping, and reducing tasks. Ahmad et al.'s [1] *Jabberwocky* framework extends this idea by adding resource management system and a high-level procedural programming language. Similarily, Noronha et al. [16] suggest a divide-and-conquer management framework inspired by corporate hierarchies. These frameworks highlight the importance of designing new programming environments but are restricted in their structural, synchronous rigidness of the underlying MapReduce programming metaphor, do not provide any explicit support of human cognitive variability [5], lack in abstractions for complex coordination patterns and task decomposition [12], and assume that computation can be fully specified ex-ante [3].

Furthermore, Zhang et al. [17] propose a system that exploits a self-organizing crowd to solve a planing under constraints problem. This system illustrates the crowd-based solution of a completely different coordination problem than the ones introduced above. It is, however, not a general-purpose approach suitable to most problems.

Complementarily, Malone et al. [13] examined about 250 different human computation systems and identified in the *Collective Intelligence Genome* the characteristics *("genes")* that can be recombined to the basic building blocks *("genome")* of human computation systems. Their conceptual classification framework suggests to characterize each building block by answering two pair of questions. First, they considered staffing *(Who is performing the task?)* and different kind of incentives *(Why are they doing it?)*. Second, they analyzed a specific system by defining the goal of a task *(What is being done?)* and problem-solving process *(How is it being done?)*. We believe that this framework is not only suitable to analyze existing applications but also to design new ones by recombining the basic building blocks.

CROWDLANG

The objective of *CrowdLang* is to build a sophisticated programming framework and language to design and execute joint collaborative computation of networked humans and machines by taking into account the respective strengths and weaknesses. In particular, *CrowdLang* incorporates explicit abstractions for group decision processes (e.g., voting and consensus mechanisms) and human computation tasks (e.g., contest, collaboration) to manage the (cognitive, error, and motivational) diversity among human agents [5]. Furthermore, and in contrast to the MapReduce inspired approaches, it supports complex coordination mechanisms (and not only bulk parallelization and loops). It also allows the modeling of of non-functional properties such as budget, completion time, or quality constraints. In a future versions, the framework will also support the *"specificity frontier"* [3] for run-time changes and task decomposition. This support of both unstructured, constraint-restricted computation as well as highly specified workflows is crucial for human computation systems because human actors have only bounded rationality and consequently ex-ante defined plans/algorithms are often imperfect or the workflow for solving a problem statement is not well-structured. The framework consists of three main components: *(1)* The *CrowdLang Library and Development Tools* simplify the design of new human computation systems. It supports the seamless reuse of existing interaction patterns by providing an extensible programming library. The integrated *intelligent assistant* supports the exploration of the whole design space through simple pattern recombination. *(2)* The *CrowdLang Engine* address the technical challenges of executing human computation algorithms by managing the crowd latency (waiting for the response by humans), debugging human computation code, and the re-executing of human computation after exceptions. The *CrowdLang Integrator* integrates different execution platforms such as micro task markets and games with a purpose.

A TEXT-TRANSLATION PROGRAM IN CROWDLANG

We illustrate the feasibility of the *CrowdLang programming framework* and its strength in interweaving networked hu-

mans and machines in a German to English text-translation task. Therefore, we developed a family of non-trivial translation programs incorporating both human and machine actors. In the development process we used the *CrowdLang Library* and *intelligent assistant* for recombining different workflow refinements. The development process included the following five steps:
(1) Identify the Core Activities: We started by defining an abstract problem-solving workflow for the translation task by identifying the abstract core activities and producer-consumer dependencies [12] among them (see Figure 1). It starts by iteratively splitting the input—an article—into paragraphs and then sentences (`Divide`). Then, the resulting sentences are processed in parallel by sequentially applying machine translation (`MT`) and crowd-based rewriting (`Rewrite`). Then the translated sentences are aggregated to paragraphs (`Aggregate`) that are then assigned to crowd-workers to improve the language quality by enhancing paragraph transitions and enforcing a consistent wording (`Improve Language Quality`). Finally, the grammatical correctness is improved by eliminating syntactical and grammatical errors (`Check Syntax`).
(2) Define the Design Space: Then, we selected four suitable interaction patterns from the *CrowdLang library* to apply them to the abstract core activities. In particular, we selected (see [15] for in detail information) *(i) Contest with Six Sigma Pruning* which uses a contest interaction pattern for generating semantical for correct sentences and improve text quality whereby the initially collected data is pruned using the six sigma rule [7, p. 320 - 330]; *(ii) Iterative Improvement* [11]; and an *(iii) Iterative Dual Pathway Structure* first presented in the context of Speech-to-Text transcription [8] for both `Rewrite` and `Improve Language Quality`. For `Check Syntax` we used an adaption of the spelling checking pattern Find-Fix-Verify [6]. *(3) Generate the Recombinations:* Next, the *CrowdLang* engine systematically generated 9 alternative refinements of the problem-solving workflow by recombining the selected patterns for the abstract core activities. *(4) Execution:* Finally, we executed these alternative refinements on a German-to-English translation task and *(5) Evaluation:* evaluated them against each other.

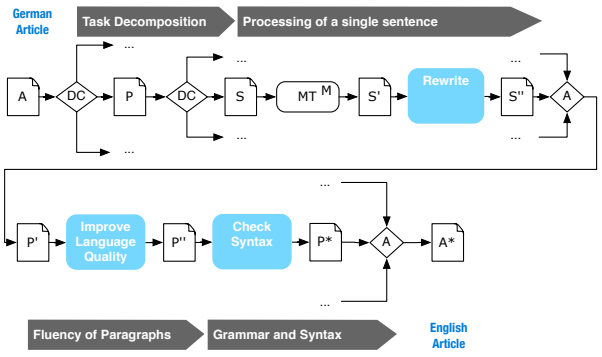

Figure 1. The abstract problem-solving workflow for the translation task in *CrowdLang* including the core activities Rewrite, Improve Language Quality, and Check Syntax.

EVALUATION

The evaluation was conducted on a standard German to English translation task. We generated translations for 15 different articles from Project Syndicate[2]— a web source of original op-ed commentaries —totaling in 153 paragraphs with 558 sentences and 10'814 words. As a baseline we considered Google Translate[3] and professional human translated gold-standard. We analyzed the resulting translations in terms of performance (throughput time, costs) and quality (adequacy, fluency, grammar).
The empirical evaluation showed that we were able to improve the quality of machine-generated translations employing 1918 monolingual crowd workers at astonishing speeds. The automatic evaluation using the METEOR score [2] showed that 2 out of 9 recombinations significantly outperformed the baseline machine translation. These two recombinations both used the Contest with Six Sigma Pruning for the `Rewrite` and Six Sigma Pruning (CPxCP) or Iterative Improvement (CPxII) for the `Improve Language Quality` subtasks repectively. Further, 283 human non-professional evaluators rated the crowd-based translations in respect to adequacy and fluency on average as 3.16 and 3.37 on an ordinal scale from 1 (Incomprehensible) to 5 (Flawless English). In comparison, the professional reference translation reached on average 4.24 and 3.58 (see Figure 2). While these results showed that the translations were far from perfect they make *useful* translations available in a fraction of the time (in average 24 minutes per article) and cost (0.09$ per sentence) of traditional solutions. The analysis of the follow-up interviews with the professional translators and of the adequacy score distribution (see [15]) showed that the differences in quality are mostly caused by a few challenges in the language structure which causes higher variance in the resulting translation. We subsequently found that installing text-improvement "subroutines" in the program to address these specific challenges can significantly improve the results by still holding the throughput time and costs low. An empirical evaluation of these subroutines are forthcoming.

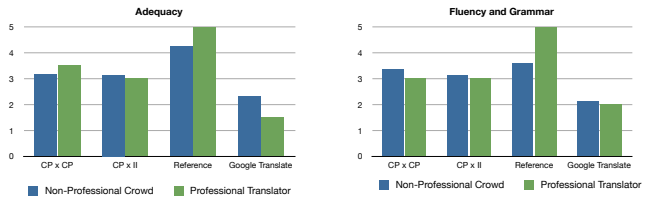

Figure 2. Mean evaluation scores for the evaluation of adequacy, fluency and grammar by 283 human non-professional evaluators and 8 professional translators

DISCUSSION AND LIMITATIONS

Our evaluation entails a number of interesting findings.
(1) The translation programs illustrate that *CrowdLang* lends itself to the simple exploration of a large design space of possible program alternatives. Whilst we cannot provide empirical proof that this feature generalizes to a large number of other approaches it does, however, indicate that a systematic exploration of the design space of possible human computation programs based on known and novel patterns may help to find good solutions. As a consequence, this technique promises to help the transition from an era of "Wizard of Oz techniques," where good functioning programs

[2]http://www.project-syndicate.org/
[3]http://translate.google.com/

are the results of lengthy trial-and-error processes, to a more engineering-oriented era.
(2) The empirical evaluation shows that it is indeed possible to significantly improve the quality of generated translations employing monolingual crowd workers at astonishing speeds. Whilst the translations are far from perfect they make useful translations available in a fraction of the time and cost of traditional solutions. We are confident that the incorporation of further text improvement "subroutines" in the program will significantly improve the result.
(3) Our adaptation of the six-sigma rule to human computation allows us to run the processes without any sophisticated pruning techniques. We could forgo any use of "control questions"– a considerable saving in terms of effort. On the downside, however, our evaluation is limited in that a usage of such quality control measures may have lead to better results. An evaluation of this question is forthcoming.
(4) Our pairing of the systematic exploration of the design space with the empirical evaluation helped us to find a novel human computation pattern CPxCP that we call *Staged Contest with Pruning*. This best performing pattern combined contests over several stages by pruning the intermediate results using the six-sigma rule and automatic comparison with the input to uncover cut-and-pastes.
As a major limitation our programs were so far only evaluated in German to English translation tasks. An evaluation using standard machine translation tasks (e.g., EU parliament dataset) as well as other language pairs is forthcoming.

CONCLUSION

In this paper we introduced human computation programming language and framework *CrowdLang*. Using the practical task of text translation we illustrated that *CrowdLang* allows the "programming" of complex human computation tasks that entail non-trivial dependencies and the systematic exploration of the design space of possible solutions via the recombination of known human computation patterns. Our empirical evaluation showed that some of the resulting programs generate "good" translations indicating that the combination of human and machine translation could provide a fruitful area of human computation. Finally, it unearthed a novel human computation pattern: "the Staged Contest with pruning". We hope that *CrowdLang* will be used by others to implement their human computation programs, as it will allow them to easily try out and compare different solutions.

REFERENCES

1. Ahmad, S., Battle, A., Malkani, Z., and Kamvar, S. The jabberwocky programming environment for structured social computing. In *Proc. of the 24th annual ACM symposium on User interface software and technology* (2011).
2. Banerjee, S., and Lavie, A. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. *Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization* (2005), 65.
3. Bernstein, A. How can cooperative work tools support dynamic group process? bridging the specificity frontier. In *Proc. of the ACM conference on Computer supported cooperative work* (2000).
4. Bernstein, A., Klein, M., and Malone, T. The process recombinator: a tool for generating new business process ideas. In *Proc. of the 20th international conference on Information Systems* (1999).
5. Bernstein, A., Klein, M., and Malone, T. Programming the global brain. *Communications of the ACM 55*, 5 (2012), 1–4.
6. Bernstein, M., Little, G., Miller, R., Hartmann, B., Ackerman, M., Karger, D., Crowell, D., and Panovich, K. Soylent: a word processor with a crowd inside. In *Proc. of the 23nd annual ACM symposium on User interface software and technology* (2010).
7. Chase, R., Aquilano, N., and Jacobs, F. *Operations management for competitive advantage*. McGraw-Hill/Irwin New York, 2006.
8. Chen, Y., Liem, B., and Zhang, H. An iterative dual pathway structure for speech-to-text transcription. In *Human Computation: AAAI Workshop* (2011).
9. Dean, J., and Ghemawat, S. Mapreduce: Simplified data processing on large clusters. *Communications of the ACM 51* (2008).
10. Kittur, A., Smus, B., Khamkar, S., and Kraut, R. Crowdforge: Crowdsourcing complex work. In *Proc. of the 24th annual ACM symposium on User interface software and technology* (2011).
11. Little, G., Chilton, L., Goldman, M., and Miller, R. Turkit: human computation algorithms on mechanical turk. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology* (2010).
12. Malone, T., and Crowston, K. The interdisciplinary study of coordination. *ACM Computing Surveys 26* (1994).
13. Malone, T., Laubacher, R., and Dellarocas, C. The collective intelligence genome. *MIT Sloan Management Review 51* (2010).
14. Malone, T., Laubacher, R., and Johns, T. General management: The age of hyperspecialization. *Harvard Business Review 89*, 7-8 (2011), 56–65.
15. Minder, P., and Bernstein, A. Crowdlang: Programming human computation systems - interweaving human and machine intelligence in a complex translation task. Tech. rep., University of Zurich, 2012.
16. Noronha, J., Hysen, E., Zhang, H., and Gajos, K. Platemate: crowdsourcing nutritional analysis from food photographs. In *Proc. of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 1–12.
17. Zhang, H., Law, E., Miller, R., Gajos, K., Parkes, D., and Horvitz, E. Human computation tasks with global constraints. CHI (2012).